APPLYING NEW OPTIMIZATION ALGORITHMS TO MODEL PREDICTIVE CONTROL

Stephen J. Wright
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439

Abstract

The connections between optimization and control theory have been explored by many researchers, and optimization algorithms have been applied with success to optimal control. The rapid pace of developments in model predictive control has given rise to a host of new problems to which optimization has yet to be applied. Concurrently, developments in optimization, and especially in interior-point methods, have produced a new set of algorithms that may be especially helpful in this context. In this paper, we reexamine the relatively simple problem of control of linear processes subject to quadratic objectives and general linear constraints. We show how new algorithms for quadratic programming can be applied efficiently to this problem. The approach extends to several more general problems in straightforward ways.

Keywords

optimization, model predictive control, interior-point methods

Introduction

In this paper we apply some recently developed techniques from the optimization literature to a core problem in model predictive control, namely, control of a linear process with quadratic objectives subject to general linear constraints. To describe our algorithms, we use the following formulation:

$$\begin{aligned} \min_{x_j,u_j} \sum_{j=0}^{N-1} \tfrac{1}{2}(x_j^T Q x_j + u_j^T R u_j) + q^T x_j + r^T u_j \\ + \tfrac{1}{2} x_N^T \tilde{Q} x_N + \tilde{q}^T x_N, \\ x_{j+1} = A x_j + B u_j, \qquad j = 0, \ldots, N-1, \qquad (1) \\ x_0 \text{ fixed}, \\ G u_j + J x_j \le g, \qquad j = 0, \ldots, N-1, \end{aligned}$$

where Q and R are positive semidefinite matrices and

$$u_j \in \mathbb{R}^m, \quad x_j \in \mathbb{R}^n, g \in \mathbb{R}^{m_c}.$$

This problem is well known from the optimal control literature, but it has been revived recently in the context of model predictive control (MPC). In MPC applications such as receding horizon control (Rawlings and Muske, 1993) and constrained linear quadratic regulation (Scokaert and Rawlings, 1995), controls are obtained by solving problems like (1) repeatedly. As we describe later, the methods we outline here can be extended easily to more general forms of (1), which may contain outputs $y_j = C x_j$, penalties on control jumps $u_{j+1} - u_j$, and so on.

The two approaches we consider in detail involve the infeasible-interior-point method and the active set method. Both methods are able to exploit the special structure in the problem (1), as they must to obtain a solution in a reasonable amount of time.

The techniques discussed here represent just one of many potential contributions that optimization can make to MPC. Developments in MPC have created a demand for fast, reliable solution of problems in which nonlinearities, noise, and constraints on the states and controls may all be present. Meanwhile, recent algorithmic developments in areas such as interior-point methods and stochastic optimization have produced powerful tools that are yet to be tested on MPC problems. By no means do we expect optimization algorithms to be a panacea for all the problems that arise in MPC. In many cases, more specialized algorithms motivated by the particular control problem at hand will be more appropriate. We do expect, however, that some MPC problems will benefit from the optimization viewpoint and that interactions between optimizers and engineers are the best way to realize these benefits.

In the next section, we sketch the way in which algorithmic research in optimization relates to applications, illustrating the point with a problem from optimal control. Next, we present the interior-point algorithm and show how it can be applied efficiently to the problem (1). We then move to the active set approach and again outline its application to (1).

Applications and Paradigms

The field of optimization was founded as a separate academic discipline during the 1940s. Its emergence was due to a number of factors. On the "demand side," there was a desire to approach the huge logistical problems posed by the wartime economy in a more systematic way, and a realization that the same techniques also could be applied to the logistical problems faced by industry and commerce during peacetime. On the "supply side," Dantzig's development of the simplex method and the appearance of digital computers were two factors that played an important role.

Connections between optimization and other mathematical disciplines, such as the calculus of variations and game theory, were recognized by the earliest researchers in the field. Today, research in optimization continues to give impetus to other areas of mathematics, such as nonsmooth analysis, linear algebra, and combinatorics. It has found applications in operations research, industrial engineering, and economics; in experimental sciences and statistics (the problem of fitting observed data to models is an optimization problem (Seber and Wild, 1989)); and in the physical sciences (for example, meteorological data assimilation (National Research Council, 1991) and superconductor modeling (Garner, Spanbauer, Benedek, Strandburg, Wright and Plassmann, 1992)).

Most researchers in optimization work with a set of standard *paradigms*, each of which is a mathematical formulation that is supposed to represent a large class of applications. Examples include linear programming, convex quadratic programming, unconstrained nonlinear optimization, and nonlinear programming. These paradigms and a few others were proposed in the early days of optimization, and they are still the focus of most of the research effort in the area. Optimization paradigms are an interface between optimization research and optimization applications. They focus the efforts of theoreticians and software developers on well-defined tasks, thereby freeing them from the effort of becoming acquainted with the details of each individual application. Linear programming is possibly the most successful paradigm of all, because a vast range of linear programs can be solved with a single piece of software, with little need for case-by-case interactions between software developers and users. More complex paradigms, such as nonlinear programming, are not so easy to apply. General software for these problems often is unable to take advantage of the special features of each instance, resulting in inefficiency. The algorithms often can be customized, however, to remove these inefficiencies.

Optimal control and model predictive control illustrate the latter point. Many problems in these areas fit into one of the standard optimization paradigms, but it is often unclear how the optimization algorithms can be applied efficiently. In some cases, special-purpose algorithms have been devised (for example, differential dynamic programming (Jacobson and Mayne, 1970)); in other cases, standard optimization algorithms such as Newton's method, the conjugate gradient method, and gradient projection algorithms have been adapted successfully to the optimal control setting (Polak, 1970; Bertsekas, 1982; Dunn and Bertsekas, 1989).

We close this section with the example of the classical discrete-time optimal control problem with Bolza objectives, a problem that arises frequently in the MPC literature (Rawlings, Meadows and Muske, 1994). This problem provides a nice illustration of the potential impact of optimization on control. It shows, too, that naive application of optimization algorithms to control problems can lead to gross inefficiencies, which can be remedied by a little customization and adaptation. The problem is

$$\begin{gathered}\min_{x_j,u_j} \textstyle\sum_{j=0}^{N-1} L_j(x_j,u_j) + \hat{L}_N(x_N), \\ x_{j+1} = f_j(x_j,u_j), \quad j=0,\dots,N-1, \quad x_0 \text{ fixed} \end{gathered} \tag{2}$$

where $x_j \in \mathbb{R}^n$, $u_j \in \mathbb{R}^m$. It can be viewed as an unconstrained optimization problem in which the unknowns are $(u_0, u_1, \dots, u_{N-1})$; the states $(x_1, x_2, \dots, x_N)$ can be eliminated through the state equations $x_{j+1} = f_j(x_j,u_j)$. However, it would be highly inefficient to solve (2) with a general implementation of Newton's method for unconstrained optimization. Such a code usually requires the user to evaluate the function, gradient, and Hessian on request at any given set of variable values. The Hessian for (2) with respect to $(u_0, \dots, u_{N-1})$ is dense, so the code would require $O(N^3m^3)$ operations simply to compute the Newton step. The paper by (Dunn and Bertsekas, 1989) shows how the same step can be obtained through a specialized calculation that takes advantage of the structure in (2) and requires only $O(N(m^3+n^3))$ operations. Hence, the Newton algorithm must be tailored to the special form of (2) if we are to have any hope of solving this problem efficiently.

The problem (2) also can be viewed as a nonlinear programming problem in which the variables are $(u_0, \dots, u_{N-1}, x_1, \dots, x_N)$ and the state equations are viewed as equality constraints. The special structure becomes transparent in this formulation, since the Jacobian of the constraints and the Hessian of the objective function are both sparse (block-banded) matrices. Therefore, a nonlinear programming code that implements some variant of sequential quadratic programming may perform quite efficiently, provided that it uses exact second derivatives and exploits the sparsity. The disadvantage is that nonlinear programming algorithms tend to have weaker global convergence properties than do unconstrained optimization algorithms. In the special case of f_j linear and L_j convex quadratic, the problem (2) is a convex programming problem, and global convergence is attained easily with either formulation.

When constraints on the controls u_j are added to (2), we still have the choice of eliminating the states x_j or not, though both formulations yield a nonlinear programming problem. (The formulation in which the states are eliminated will have simpler constraints and a more complicated objective function.) When constraints on the states are introduced, however, elimination of the x_j becomes more problematic, and there is little choice but to view the problem as a nonlinear programming problem in which the unknowns are $(u_0, u_1, \ldots, u_{N-1}, x_1, x_2, \ldots, x_N)$. We consider the linear-quadratic form of this problem in the next two sections.

Interior-Point Methods for Linear-Quadratic Problems

In this section, we consider the linear-quadratic problem (1). It has frequently been noted that this problem is, in optimization terms, a convex quadratic program. Two successful methods for addressing this class of problems are the active set method described by (Fletcher, 1987) and the interior-point method described by (Wright, 1996). However, the special structure of (1) means that we must take care in applying either approach to this problem. A naive application of a quadratic programming code based on the active-set method (for instance, QPOPT (Gill, Murray, Saunders and Wright, 1991)) will give poor results, typically requiring $O(N^3(m+n+m_c)^3)$ operations, where m_c is the number of rows in the matrices G and J. In this section, we show how the interior-point algorithm can be applied to (1), while in the next section we examine the adaptations that are needed to make the active set approach more efficient.

We state at the outset that the interior-point approach we describe here can be adapted to various modifications and generalizations of (1) without significant loss of efficiency. For instance,

- the matrices Q, R, A, B, G, and J can vary with j;
- an output vector $y_j = Cx_j$ can be incorporated into the objective and constraints;
- we can incorporate constraints and objective terms that involve states/controls from adjacent stages; for example, a penalty on the control move $(u_{j+1} - u_j)_i$ for some component $i = 1, 2, \ldots, m$.

The last generalization is useful when the problem is obtained as a discretization of the continuous problem, since many discretization schemes for ordinary differential equations and differential algebraic equations lead to relationships between states and controls at a number of adjacent time points.

The rest of this section is organized as follows. We define the *mixed monotone linear complementarity problem* (mLCP), a powerful paradigm that generalizes the optimality conditions for linear and quadratic programs and is a convenient platform for describing interior-point methods. We then outline an infeasible-interior-point algorithm for the mLCP and discuss its properties. Finally, we customize this algorithm to convex quadratic programming and the linear-quadratic problem (1).

Mixed Linear Complementarity and the Infeasible-Interior-Point Framework

The mLCP is defined in terms of a square, positive semidefinite matrix $M \in \mathbb{R}^{n\times n}$ and a vector $q \in \mathbb{R}^n$. The problem is to find vectors z, x, and s such that

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} z \\ x \end{bmatrix} + \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 0 \\ s \end{bmatrix}, \qquad (3)$$

$$x \geq 0, \; s \geq 0, \; x^T s = 0. \qquad (4)$$

Here, M_{11} and M_{22} are square submatrices of M with dimensions n_1 and n_2, respectively, and the vector q is partitioned accordingly.

The infeasible-interior-point algorithm for (3),(4) starts at point (z^0, x^0, s^0) for which $x^0 > 0$ and $s^0 > 0$ (*interior* to the nonnegative orthant) but possibly *infeasible* with respect to the constraints (3). All iterates (z^k, x^k, s^k) retain the positivity properties $x^k > 0$ and $s^k > 0$, but the infeasibilities and the *complementarity gap* defined by

$$\mu_k = (x^k)^T s^k / n_2 \qquad (5)$$

are gradually reduced to zero as $k \to \infty$. Each step of the algorithm is a modified Newton step for the system of nonlinear equations defined by the feasibility conditions (3) and the complementarity conditions $x_i s_i = 0$, $i = 1, 2, \ldots, n_2$. We can write this system as

$$F(z,x,s) \stackrel{\text{def}}{=} \begin{bmatrix} M_{11}z + M_{12}x + q_1 \\ M_{21}z + M_{22}x - s + q_2 \\ XSe \end{bmatrix} = 0, \qquad (6)$$

where we have used the notation

$$X = \operatorname{diag}(x_1, x_2, \ldots, x_{n_2}), \qquad S = \operatorname{diag}(s_1, s_2, \ldots, s_{n_2}).$$

The algorithm has the following form:

```
Algorithm IIP
Given $(z^0, x^0, s^0)$ with $(x^0, s^0) > 0$;
for $k = 0, 1, 2, \ldots$
    for some $\sigma_k \in (0,1)$, solve
```

$$\begin{bmatrix} M_{11} & M_{12} & 0 \\ M_{21} & M_{22} & -I \\ 0 & S^k & X^k \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_1^k \\ -r_2^k \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix}, \qquad (7)$$

```
    to obtain $(\Delta z^k, \Delta x^k, \Delta s^k)$, where
```

$$\begin{aligned} r_1^k &= M_{11} z^k + M_{12} x^k + q_1, \\ r_2^k &= M_{21} z^k + M_{22} x^k - s^k + q_2, \\ e &= (1, 1, \ldots, 1)^T. \end{aligned}$$

```
    set
```

$$\begin{aligned} &(z^{k+1}, x^{k+1}, s^{k+1}) \\ &\quad = (z^k, x^k, s^k) + \alpha_k (\Delta z^k, \Delta x^k, \Delta s^k), \end{aligned} \tag{8}$$

```
    for some $\alpha_k \in (0,1]$ that retains
        $(x^{k+1}, s^{k+1}) > 0$;
end(for).
```

Note that (7) differs from the pure Newton step for (6) only because of the term $\sigma_k \mu_k e$ on the right-hand side. This term plays a stabilizing role, ensuring that the algorithm converges steadily to a solution of (3),(4) while remaining inside the positive orthant defined by $(x, s) > 0$.

The only two parameters to choose in implementing Algorithm IIP are the scalars σ_k and α_k. The convergence analysis leaves the choice of σ_k relatively unfettered (it is often confined to the range $[\sigma, 0.8]$, where σ is a fixed parameter, typically 10^{-3}), but α_k is required to satisfy the following conditions.

(i) The reduction factor for the infeasibility norms $\|r_1\|$ and $\|r_2\|$ should be smaller than the reduction factor for μ; that is, the ratios $\|r_1^k\|/\mu_k$ and $\|r_2^k\|/\mu_k$ should decrease monotonically with k;

(ii) The pairwise products $x_i s_i$, $i = 1, \ldots, n_2$ should all approach zero at roughly the same rate; that is, the ratios $x_i^k s_i^k / \mu_k$ should remain bounded away from zero for all i and all k. (Note that μ_k represents the average values of the terms $x_i^k s_i^k$.)

(iii) We require sufficient decrease in μ, in the sense that the decrease actually obtained is at least a small fraction of the decrease predicted by the linear model (7). (See (Dennis and Schnabel, 1983) for a discussion of sufficient decrease conditions.)

(iv) The chosen value of α_k should not be too much smaller than the largest possible value for which (i)–(iii) are satisfied.

For details, see (Wright, 1993b; Wright, 1996). When σ_k and α_k satisfy these conditions, global convergence to a solution of (3),(4) is attained whenever such a solution exists. A slightly enhanced version of the algorithm, in which σ_k is allowed to be zero on some of the later iterations, exhibits superlinear convergence under some additional assumptions. The property that excites many of the theoreticians working on interior-point methods—polynomial complexity—is also attained when the starting point (z^0, x^0, s^0) has sufficiently large x^0 and s^0 components, relative to the initial residuals r_1^0 and r_2^0 and the solutions of (3),(4).

In practical implementations of Algorithm IIP, α_k often is chosen via the following simple heuristic. First, we set $\alpha_k^{\max}$ to be the supremum of the following set:

$$\{\alpha \in (0,1] \,|\, (z^k, x^k, s^k) + \alpha(\Delta z^k, \Delta x^k, \Delta s^k) > 0\}. \tag{9}$$

Then we set

$$\alpha_k = \min(1, 0.995 * \alpha_k^{\max}). \tag{10}$$

That is, we forget about the theoretical conditions (i)–(iv) above and simply choose α_k to step almost all the way to the boundary of the nonnegative orthant. This tension between theory and practice existed for a long time during the development of primal-dual methods. However, recent work has reconciled the differences. There exist relaxed versions of conditions (i)–(iv) that are satisfied by the "practical" choice of α_k from (9),(10). Hence, the parameters σ_k and α_k and be chosen to make Algorithm IIP both practically efficient and theoretically rigorous.

The major operation to be performed at each step of Algorithm IIP is the solution of the linear system (7). The matrix in this system obviously has a lot of structure due to the presence of the zero blocks and the diagonal components I, S^k, and X^k. Additionally, the matrix M is sparse in most cases of practical interest, including our motivating problem (1), so sparse matrix factorizations are called for. In general, these are fairly complex pieces of software, but problems of the form (1) require only banded factorization code, which is comparatively simple.

The first step in solving (7) is to eliminate the Δs component. Since the diagonal elements of X^k are positive, we can rearrange the last block row in (7) to obtain

$$\begin{aligned} \Delta s &= (X^k)^{-1}(-X^k S^k e + \sigma_k \mu_k e - S^k \Delta x^k) \\ &= -s_k + (X^k)^{-1}(\sigma_k \mu_k e - S^k \Delta x^k). \end{aligned}$$

By substituting into the first two rows of (7), we obtain

$$\begin{aligned} &\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} + (X^k)^{-1} S^k \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta x \end{bmatrix} \\ &\quad = \begin{bmatrix} -r_1^k \\ -r_2^k - s_k + \sigma_k \mu_k (X^k)^{-1} e \end{bmatrix}. \end{aligned} \tag{11}$$

In most cases, some of the partitions M_{11}, M_{12}, M_{21}, or M_{22} are zero or diagonal or have some other simple structure, so further reduction of the system (11) is usually possible. This phenomenon happens, for instance, when (3),(4) is derived from a linear or quadratic program, as we show below.

Since the factorization of the coefficient matrix in (7) comprises most of the work at each iteration, we may be led to ask whether it is really necessary to compute a fresh factorization every time. A set of heuristics in which the factorization is essentially re-used on alternate steps was proposed by (Mehrotra, 1992). Mehrotra's algorithm has proved to be successful in practice and is the basis for the vast majority of interior-point codes for linear programming.

Linear and Quadratic Programming as mLCPs

We now show how linear and convex quadratic programming problems can be expressed in the form (3),(4) and solved via Algorithm IIP. Consider first the linear program in standard form:

$$\min_x c^T x \text{ subject to } Ax = b,\ x \geq 0, \tag{12}$$

where c and x are vectors in R^n, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m\times n}$. The dual of (12) is

$$\max_{\lambda,s} b^T\lambda \text{ subject to } A^T\lambda + s = c,\ s \geq 0, \tag{13}$$

where $\lambda \in \mathbb{R}^m$ are the dual variables (or, alternatively, the Lagrange multipliers for the constraints $Ax = b$) and $s \in \mathbb{R}^n$ are the dual slacks. The Karush-Kuhn-Tucker (KKT) conditions for (12),(13) are as follows:

$$\begin{aligned} A^T\lambda + s &= c, \\ Ax &= b, \\ x \geq 0,\ s \geq 0, \qquad & x^T s = 0. \end{aligned} \tag{14}$$

Because (12) is a convex programming problem, the KKT conditions are both necessary and sufficient. Hence, we can find a primal-dual solution for the linear program by finding a vector (x, λ, s) that satisfies the conditions (14). We can verify that (14) has the form (3),(4) by making the following identifications between these two systems:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} 0 & A \\ -A^T & 0 \end{bmatrix},\ \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} -b \\ c \end{bmatrix},$$
$$z \leftarrow \lambda,\ x \leftarrow x,\ s \leftarrow s.$$

Hence the KKT conditions (14) are an mLCP, and we can obtain solutions to the linear program (12) and its dual (13) simultaneously by applying Algorithm IIP to (14).

Next, we consider the following general convex quadratic program:

$$\min_z \tfrac{1}{2} z^T Q z + c^T z \text{ s.t. } Hz = h,\ Gz \leq g, \tag{15}$$

where Q is a symmetric positive semidefinite matrix. The KKT conditions for this system are

$$\begin{aligned} Qz + H^T\zeta + G^T\lambda &= -c, \\ -Hz &= -h, \\ -Gz - t &= -g, \\ t \geq 0,\ \lambda \geq 0, \qquad & t^T\lambda = 0. \end{aligned} \tag{16}$$

The following identifications confirm that the system (15) is an mLCP:

$$M_{11} = \begin{bmatrix} Q & H^T \\ -H & 0 \end{bmatrix},\ M_{12} = \begin{bmatrix} G^T \\ 0 \end{bmatrix},$$
$$M_{21} = \begin{bmatrix} -G & 0 \end{bmatrix},\ M_{22} = 0,$$
$$q_1 = \begin{bmatrix} c \\ h \end{bmatrix},\ q_2 = g,$$
$$z \leftarrow \begin{bmatrix} z \\ \zeta \end{bmatrix},\ x \leftarrow \lambda,\ s \leftarrow t.$$

The reduced form (11) of the linear system to be solved at each iteration of Algorithm IIP is

$$\begin{bmatrix} Q & H^T & G^T \\ -H & 0 & 0 \\ -G & 0 & (\Lambda^k)^{-1}T^k \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\zeta \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -r_c^k \\ -r_h^k \\ -r_g^k - t^k + \sigma_k\mu_k(\lambda^k)^{-1}e \end{bmatrix}. \tag{17}$$

Here, μ_k is defined as $\mu_k = (t^k)^T\lambda^k/m$, where m is the number of inequality constraints in (15). It is customary to multiply the last two block rows in (17) by -1, so that the coefficient matrix is symmetric indefinite. We then obtain

$$\begin{bmatrix} Q & H^T & G^T \\ H & 0 & 0 \\ G & 0 & -(\Lambda^k)^{-1}T^k \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\zeta \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -r_c^k \\ r_h^k \\ r_g^k + t^k - \sigma_k\mu_k(\lambda^k)^{-1}e \end{bmatrix}. \tag{18}$$

Since $(\Lambda^k)^{-1}T^k$ is diagonal with positive diagonal elements, we can eliminate $\Delta\lambda$ from (18) to obtain an even more compact form:

$$\begin{bmatrix} Q + G^T\Lambda^k(T^k)^{-1}G & H^T \\ H & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\zeta \end{bmatrix} = \begin{bmatrix} -r_c^k + G^T[\Lambda^k(T^k)^{-1}r_g^k + \lambda^k - \sigma_k\mu_k(T^k)^{-1}e] \\ r_h^k \end{bmatrix}. \tag{19}$$

Factorizations of symmetric indefinite matrices have been studied extensively in the linear algebra literature; see (Golub and Van Loan, 1989) for references to the major algorithms. Standard software is available, at least in the dense case (Anderson, Bai, Bischof, Demmel, Dongarra, Du Croz, Greenbaum, Hammarling, McKenney, Ostrouchov and Sorensen, 1992). Algorithms for the sparse case have also received considerable attention recently (Ashcraft, Grimes and Lewis, 1995) and have been implemented in the context of interior-point methods by (Fourer and Mehrotra, 1993).

In solving the linear systems (18) or (19), we are free to reorder the rows and columns of the coefficient matrix in any way we choose, before or during the factorization. As we see below, the problem (1) benefits dramatically from such a reordering, since the coefficient matrix in this case becomes a narrow-banded matrix, which is easily factored via existing linear algebra software such as LAPACK (Anderson et al., 1992).

"Soft" Constraints and Penalty Terms

The cost of introducing slack variables and dummy variables into the formulation (15) is often surprisingly small when the quadratic program is solved by the technique outlined above, even though the total number of variables in the problem may increase appreciably. The reason is that the new variables can often be substituted out of the linear system (11), as in (18) and (19), so that they may not affect the size of the linear system that we actually solve. This comment is relevant when we are adding norm constraints or "soft" constraints to the problems (15) or (1), as in (Scokaert and Rawlings, 1996). Suppose, for instance, that we wish to include the soft constraint $G_s z \leq g_s$ in (15), and we choose to do this by including a term $\nu \|(G_s z - g_s)_+\|_1$ to the objective function. (The subscript "+" denotes the positive part of a vector, obtained by replacing its negative components by zeros.) To restore the problem to the form of a standard convex quadratic program, we introduce the "dummy" vector v, and

- add the term $\nu e^T v$ to the objective, where $e = (1, 1, \ldots, 1)^T$;
- introduce the additional constraints $v \geq 0$, $v \geq G_s z - g_s$.

By computing the mLCP form (3),(4) of the expanded problem, applying Algorithm IIP, and reducing its step equations as far as we can via simple substitutions, we find that the linear system is ultimately no larger than (19). The details are messy and we omit them from this discussion.

In the case of soft constraints and penalty terms, the consequence of this observation is that the amount of work per iteration is not really sensitive to whether we choose a 1-norm penalty or an ∞-norm penalty, though the latter option adds fewer dummy variables to the formulation of the problem.

Solving LQR Efficiently via the mLCP Formulation

The linear-quadratic problem (1) is obviously a special case of (15), as we see by making the following identifications between the data:

$$Q \leftarrow \begin{bmatrix} R & & & & & & \\ & Q & & & & & \\ & & R & & & & \\ & & & \ddots & & & \\ & & & & Q & & \\ & & & & & R & \\ & & & & & & \tilde{Q} \end{bmatrix},$$

$$G \leftarrow \begin{bmatrix} G & J & & & & \\ & & G & J & & \\ & & & \ddots & \ddots & \\ & & & & G & J \end{bmatrix},$$

$$H \leftarrow \begin{bmatrix} B & -I & & & & \\ & A & B & -I & & \\ & & & \ddots & & \\ & & & A & B & -I \end{bmatrix},$$

$$z \leftarrow \begin{bmatrix} u_0 \\ x_1 \\ u_1 \\ \vdots \\ u_{N-1} \\ x_N \end{bmatrix}, \quad c \leftarrow \begin{bmatrix} r \\ q \\ r \\ \vdots \\ r \\ \tilde{q} \end{bmatrix},$$

$$g \leftarrow \begin{bmatrix} g \\ g \\ \vdots \\ g \end{bmatrix}, \quad h \leftarrow \begin{bmatrix} -Ax_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

As suggested earlier, the matrices in this problem are all *block-banded*; their nonzeros are clustered around a line connecting the upper-left and lower-right corners of the matrix. When we formulate the linear system (17), the nonzeros seem to become widely dispersed. However, by interleaving the states x_j, the controls u_j, the adjoints p_j and the Lagrange multipliers λ_j for the constraints $Gu_j + Jx_{j+1} \leq g$, we can return the coefficient matrix in (17) to banded form. We order the unknowns in the system as

$$(\Delta u_0, \Delta\lambda_0, \Delta p_1, \Delta x_1, \Delta u_1, \ldots, \Delta\lambda_{N-1}, \Delta p_N, \Delta x_N)$$

and rearrange the rows and columns of the matrix accordingly to obtain

$$\begin{bmatrix} R & G^T & B^T & & & & & & \\ G & -D_0^k & 0 & & & & & & \\ B & 0 & 0 & -I & & & & & \\ & & -I & Q & 0 & J^T & A^T & & \\ & & & 0 & R & G^T & B^T & & \\ & & & J & G & -D_1^k & 0 & & \\ & & & A & B & 0 & 0 & -I & \\ & & & & & & -I & Q & \ddots \\ & & & & & & & \ddots & \ddots \end{bmatrix}. \tag{20}$$

(We have used D_j^k to denote sections of the diagonal matrix $(\Lambda^k)^{-1}T^k$ from (17).) This system can be reduced further by using the diagonal entries D_j^k to eliminate $\Delta\lambda_j$, $j = 0, 1, \ldots, N-1$.

The computational savings obtained by recovering bandedness are significant. If we assume the dimensions

$$u_j \in \mathbb{R}^m, \quad x_j \in \mathbb{R}^n, \quad \lambda_j \in \mathbb{R}^{m_c},$$

the banded matrix has total dimension $N(2n+m+m_c)$ and half-bandwidth of $(2n+m+m_c-1)$. Therefore, the time required to factor this matrix is $O(N(2n+m+m_c)^3)$, compared with $O(N^3(2n+m+m_c)^3)$ for a dense matrix of the same size. In the absence of constraints (that is, $m_c = 0$), this cost has exactly the same order as the cost of solving a linear-quadratic version of (2) by using dynamic programming techniques.

Stagewise ordering of the equations and variables in (17) is the key to obtaining a banded structure. As mentioned above, we can maintain the banded structure when outputs $y_k = Cx_k$ and other constraints are introduced into the model, provided that we continue to order the variables by stage.

The techniques outlined above are quite similar to those described in (Wright, 1993a). The difference lies in the use of infeasible-interior-point methods above, in contrast to the techniques of (Wright, 1993a) which combined feasible interior-point methods with an embedding of the original problem into an expanded problem for which a feasible initial point is easy to compute. The new approach is cleaner, more practical, and more efficient, while remaining theoretically rigorous.

Active Set Methods for Linear-Quadratic Problems

The structure of the problem (1) can also be exploited when we use an active set method in place of the interior-point method described above. Again, we find that the linear algebra at each step can be performed in terms of banded matrices rather than general dense matrices. The details are different from (and somewhat more complicated than) the interior-point case. We start by sketching a single iterate of the active set approach. For a more complete description, see (Fletcher, 1987).

Active set methods for the general convex program (15) generate a sequence of feasible iterates. At each iterate, a certain subset of the constraints $Gz \le g$ are *active* (that is, hold as equalities). On each step, we choose a subset of the active set known as the *working set*. (Typically, the working set either is identical to the active set or else contains just one less constraint.) We then compute a step from the current point z that minimizes the objective function in (15) while maintaining activity of the constraints in the working set, and also ensuring that the original equality constraints $Hz = h$ remain satisfied. If we denote by $\bar{G}$ the subset of rows of G that make up the working set, the step Δz is obtained by solving the following system:

$$\begin{gathered} \min_{\Delta z} \ \tfrac{1}{2}(z+\Delta z)^T Q (z+\Delta z) + c^T(z+\Delta z) \\ \text{subject to } H\Delta z = 0, \quad \bar{G}\Delta z = 0. \end{gathered}$$

Equivalently, we have

$$\min_{\Delta z} \ \tfrac{1}{2}\Delta z^T Q \Delta z + \tilde{c}^T \Delta z, \ \text{ s.t. } \ H\Delta z = 0, \quad \bar{G}\Delta z = 0, \tag{21}$$

where $\tilde{c} = c + Qz$. The KKT conditions for (21) are that Δz is a solution of this system if and only if there are vectors $\Delta\zeta$ and $\Delta\bar{\lambda}$ such that $(\Delta z, \Delta\zeta, \Delta\bar{\lambda})$ satisfies the following system:

$$\begin{bmatrix} Q & H^T & \bar{G}^T \\ H & 0 & 0 \\ \bar{G} & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\zeta \\ \Delta\bar{\lambda} \end{bmatrix} = \begin{bmatrix} -\tilde{c} \\ 0 \\ 0 \end{bmatrix}. \tag{22}$$

We can obtain Δz from this system and then do a line search along this direction, stopping when a new constraint is encountered or when the minimum of the objective function along this direction is reached.

Note the similarity between (22) and (17). The coefficient matrices differ only in that there is no diagonal term in the lower left of (22), and some rows are deleted from G. For the problem (1), the matrices Q, H, and $\bar{G}$ are banded, so a stagewise reordering of the rows and columns in (22) again produces a banded system. The matrix will not be quite as regular as (20), because of the missing columns in $\bar{G}$, but similar savings can be achieved in factoring it.

The banded matrix is best factored by Gaussian elimination with row partial pivoting, as implemented in the LAPACK routine `dgbtrf` and its affiliated solution routine `dgbtrs` (Anderson et al., 1992). To the author's knowledge, there is no software that can exploit the fact that the matrix is symmetric in addition to being banded.

Updating Factorizations

We have noted that the matrix in (22) is banded for the problem (1), so we can use software tailored to such matrices to obtain significant savings over the dense linear algebra that is usually employed in standard quadratic programming software. The story is not quite this simple, however. The systems (22) that we solve at each iteration are closely related to one another, differing only in that a column is added and/or deleted from $\bar{G}$. Therefore, it does not make sense to solve each system "from scratch"; we try instead to modify the matrix factorization that was computed at an earlier iteration to accommodate the minor changes that have occurred since then. In the general case of dense matrices, updating of factorizations has been studied extensively and is implemented in software for many kinds of optimization problems. (Simplex algorithms for linear programming, for instance, depend heavily on efficient updating of the basis matrix at each iteration.) The question we face here is: Can we perform efficient updating of the factorization while still exploiting bandedness? We answer this question in the affirmative by sketching a technique for re-solving (22) after a row has been added to or deleted from $\bar{G}$.

We start with addition of a column to the system (22). Changing the notation for convenience, we denote the original coefficient matrix by M and the new row/column by a. We assume without loss of generality that a is ordered last. Then the updated matrix $\bar{M}$ is

$$\bar{M} = \begin{bmatrix} M & a \\ a^T & 0 \end{bmatrix}.$$

We assume that the LU factorization of the original matrix M is known, so that there is a permutation matrix P, a lower triangular L and upper triangular U such that

$$PM = LU. \tag{23}$$

(If M is banded, then the factors L and U also have nonzeros only in locations near their diagonals.) We can easily modify L and U to accommodate the new row/column by adding an extra row and column to each factor to obtain

$$\begin{bmatrix} P & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} M & a \\ a^T & 0 \end{bmatrix} = \begin{bmatrix} L & \\ a^T U^{-1} & 1 \end{bmatrix} \begin{bmatrix} U & L^{-1}Pa \\ 0 & \alpha \end{bmatrix},$$

where

$$\alpha = -a^T U^{-1} L^{-1} P a.$$

Hence, the factorization can be updated at the cost of triangular substitution with each of L and U. For (1), the cost of this process is $O(N(m+n+m_c)^2)$, so it is less expensive than refactoring $\bar{M}$ from scratch, unless $(m+n+m_c)$ is very small. Since the new row/column does not participate in the pivoting, the new row of L and column of U are dense in general, so the factors are not as sparse as they would be in a factorization from scratch. Stability issues may arise, since the diagonal element α may be small. A good strategy for dealing with these problems is to compute a fresh factorization whenever α becomes too small by some measure, or when $O(m+n+m_c)$ iterations have passed since the last refactorization.

We turn now to the case in which a row and column are deleted from M. We assume that the factorization (23) is known and that we obtain a new coefficient matrix $\hat{M}$ by deleting row i and column j from the matrix PM. (Note that the indices of the deleted row and column will be identical in the original matrix M by symmetry, but row pivoting may cause them to differ in the permuted matrix PM.) Obviously, we can modify L and U so that their product equals $\hat{M}$ by simply deleting the ith row of L and the jth column of U. Unfortunately, these deletions cause L and U to become nontriangular; the entries $L_{i+1,i+1}, L_{i+2,i+2}, \ldots$ now appear *above* the diagonal of the modified version of L. We can restore triangularity by removing the ith *column* of L as well as the ith row, and similarly removing the jth *row* of U as well as the jth column. The modified matrix $\hat{M}$ can be expressed in terms of these modified L and U factors as follows:

$$\hat{M} = \hat{L}\hat{U} + vw^T, \tag{24}$$

where $\hat{L}$ is the matrix L with ith row and column removed and $\hat{U}$ is U with the jth row and column removed, and we have

$$v = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ L_{i+1,i} \\ \vdots \\ L_{n,i} \end{bmatrix}, \qquad w = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ U_{j,j+1} \\ \vdots \\ U_{j,n} \end{bmatrix}.$$

The expression (24) shows that the product $\hat{L}\hat{U}$ of two triangular matrices differs from $\hat{M}$ by a rank-one matrix, which can be accounted for by using the Sherman-Morrison-Woodbury formula (Golub and Van Loan, 1989, page 51). From this expression, we have

$$\begin{aligned} \hat{M}^{-1} &= (\hat{L}\hat{U} + vw^T)^{-1} \\ &= (\hat{L}\hat{U})^{-1} + \frac{[(\hat{L}\hat{U})^{-1}v][w^T(\hat{L}\hat{U})^{-1}]}{1 + w^T(\hat{L}\hat{U})^{-1}v}. \end{aligned}$$

Hence, the solution $\hat{x}$ of the system

$$\hat{M}\hat{x} = \hat{c}$$

can be written as

$$\begin{aligned} \hat{x} &= \hat{M}^{-1}\hat{c} \\ &= (\hat{L}\hat{U})^{-1}\hat{c} + \frac{(\hat{L}\hat{U})^{-1}v}{1 + w^T(\hat{L}\hat{U})^{-1}v} w^T(\hat{L}\hat{U})^{-1}\hat{c}. \end{aligned}$$

In computing $\hat{x}$ via this formula, the main operations are to perform two pairs of triangular substitutions with $\hat{L}$ and $\hat{U}$: one pair to compute $(\hat{L}\hat{U})^{-1}\hat{c}$ and another to compute $(\hat{L}\hat{U})^{-1}v$.

Hot Starts

In MPC, the problem (1) is not usually encountered in isolation. On the contrary, we usually need to solve a sequence of these problems in which the data A, B, Q, etc, and/or the starting point x_0 vary only slightly from one problem to the next. It is highly desirable that the algorithms should be able to take advantage of this fact. The information could be used, for instance, to choose good starting values for all the variables or to make a good guess of the active constraint set (that is, the matrix $\bar{G}$ in (21)). The process of using this information is called *hot starting*.

Sequences of similar linear-quadratic problems can arise in the control of nonlinear systems. When we apply the sequential quadratic programming algorithm to a constrained version of (2), we obtain a search direction at each iteration by solving a problem like (1). (Actually, we solve a slightly more general problem in which the data A, B, Q, R varies with stage index j and linear terms may appear in the objectives and constraints.) As the iterates converge to a solution of the nonlinear problem, the data matrices become more and more similar from one iteration to the next. A starting guess of $u_j \equiv 0$ and $x_j \equiv 0$ is best, because the subproblem is obtained by approximating the nonlinear problem around the current iterate. We can however make an excellent guess at the active set and the initial working set, particularly on later iterations. Active set methods will typically require just a few steps to identify the correct active set from this good initial guess.

Model predictive control also gives rise to sequence of similar linear-quadatic problems. The usual procedure is to solve a problem like (1) using the current state of the system as the initial value x_0, and then apply the control u_0 until the next time point is reached. The process is then repeated (Scokaert and Rawlings, 1995; Rawlings and Muske, 1993). In the absence of disturbances, the problems (1) are very similar on successive steps, sometimes differing only in the initial value x_0. An excellent starting point can be obtained from the solution at the previous set by setting

$$\begin{aligned}(x_0, x_1, \ldots, x_N) &= (x_0^{\text{new}}, x_2^-, \ldots, x_N^-, x_N^{\text{new}}),\\ (u_0, u_1, \ldots, u_{N-1}) &= (u_1^-, u_2^-, \ldots, u_{N-2}^-, u_{N-1}^{\text{new}}),\end{aligned}$$

where u_j^-, x_j^- are the solution components at the previous step, x_0^{new} is the new initial state, and x_N^{new} and u_{N-1}^{new} are some well-chosen estimates for the final stages. In the case of the active set approach, an excellent starting guess can also be made for the active constraint matrix $\bar{G}$.

In some situations, particularly when disturbances are present, the starting point chosen by the obvious techniques may not be feasible even though it is close to a solution. This represents no problem for the infeasible-interior-point approach, although it is desirable in Algorithm IIP for the initial complementarity to be comparable in size to the initial infeasibilities. The active set method assumes a feasible starting point. One remedy is to use a two-phase approach, in which we solve a "Phase I" problem to find a feasible point, then a "Phase II" problem to find the optimum. A second option is to introduce penalty terms into the objective for the infeasibilities and then obtain a solution in a single phase (provided that a heavy enough penalty is imposed.)

Active set methods typically gain more from hot starting than do interior-point methods, for reasons that are not yet fully understood. On linear programming problems, the best interior-point codes gain about a factor of three in compute time when they are hot started, in comparison with a "cold" (i.e., no prior information) start. The relative savings for simplex/active set methods are significantly higher. It is difficult to predict how much the situation will change when we consider the problem class (1). Numerical testing is the only way to find out.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. and Sorensen, D. (1992). *LAPACK User's Guide*, SIAM, Philadelphia.

Ashcraft, C., Grimes, R. L. and Lewis, J. G. (1995). Accurate symmetric indefinite linear equation solvers. in preparation.

Bertsekas, D. P. (1982). Projected Newton methods for optimization problems with simple constraints, *SIAM Journal on Control and Optimization* **20**: 221–246.

Dennis, J. E. and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization*, Prentice-Hall, Englewood Cliffs, NJ.

Dunn, J. C. and Bertsekas, D. P. (1989). Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems, *Journal of Optimization Theory and Applications* **63**: 23–38.

Fletcher, R. (1987). *Practical Methods of Optimization*, second edn, John Wiley and Sons, New York.

Fourer, R. and Mehrotra, S. (1993). Solving symmetric indefinite systems in an interior-point method for linear programming, *Mathematical Programming* **62**: 15–39.

Garner, J., Spanbauer, M., Benedek, R., Strandburg, K. J., Wright, S. J. and Plassmann, P. E. (1992). Critical fields of josephson-coupled superconducting multilayers, *Physical Review B* **45**: 7973–7983.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1991). Inertia-controlling methods for general quadratic programming, *SIAM Review* **33**: 1–36.

Golub, G. H. and Van Loan, C. F. (1989). *Matrix Computations*, 2nd edn, The Johns Hopkins University Press, Baltimore.

Jacobson, D. H. and Mayne, D. Q. (1970). *Differential Dynamic Programming*, American Elsevier, New York.

Kall, P. and Wallace, S. W. (1994). *Stochastic Programming*, John Wiley and Sons.

Mehrotra, S. (1992). On the implementation of a primal-dual interior point method, *SIAM Journal on Optimization* **2**: 575–601.

National Research Council (1991). *Four-Dimensional Model Assimilation of Data*, National Academy Press.

Polak, E. (1970). *Computational Methods in Optimization*, Academic Press, New York.

Rawlings, J. B., Meadows, E. S. and Muske, K. R. (1994). Nonlinear model predictive control: A tutorial and survey, *Proceedings of ADCHEM*, Tokyo, Japan.

Rawlings, J. B. and Muske, K. R. (1993). The stability of constrained receding horizon control, *IEEE Transactions on Automatic Control* **38**(10): 1512–1516.

Scokaert, P. O. M. and Rawlings, J. B. (1995). Constrained linear quadratic regulation, *Technical report*, Department of Chemical Engineering, University of Wisconsin-Madison.

Scokaert, P. O. M. and Rawlings, J. B. (1996). On infeasibilities in model predictive control, *Chemical Process Control: CPC-V*, CACHE.

Seber, G. A. F. and Wild, C. J. (1989). *Nonlinear Regression*, John Wiley and Sons, New York.

Vandenberghe, L. and Boyd, S. (1994). Semidefinite programming, *Technical report*, Electrical Engineering Department, Stanford University, Stanford, CA 94305. To appear in SIAM Review.

Wright, S. J. (1993a). Interior point methods for optimal control of discrete-time systems, *Journal of Optimization Theory and Applications* **77**: 161–187.

Wright, S. J. (1993b). A path-following interior-point algorithm for linear and quadratic optimization problems, *Preprint MCS-P401-1293*, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill. To appear in Annals of Operations Research.

Wright, S. J. (1996). Primal-Dual Interior-Point Methods. Book in preparation. See URL http://www.mcs.anl.gov/home/wright/ippd.html.